3Р

УДК 004.891

В.М. Рувинская[1], Е.Л. Беркович[2], А.А. Лотоцкий[2]
[1]Одесский национальный политехнический университет, Украина
[2]Одесский национальный университет им. Мечникова, Украина
iolnlen@te.net.ua, for_lotos@mail.ru

Эвристические методы детектирования вредоносных программ на основе сценариев

В статье рассмотрены существующие методы борьбы с вредоносными программами, а также предложен интеллектуальный метод их детектирования на основе сценариев. Сценарии позволяют представить поведение вредоносных программ в иерархическом виде; приведен пример сценария поведения паразитического вируса и сценария внедрения кода в систему. Предложено формально описывать сценарии на основе регулярных выражений. Разработана архитектура эвристика, содержащего экспертную систему на основе сценариев.

Введение

В современном информационном мире уже давно наступил момент, когда стоимость информации значительно превзошла стоимость систем для её обработки и хранения. На сегодняшний день уже, вероятно, не осталось компаний, которые бы не вели весь свой документооборот в электронном виде, пользователи повсеместно используют номера своих кредитных карт, пароли и другие частные данные. Всё это является желанной целью преступников [1].

По цели воздействия на компьютерные системы различают три основных типа угроз безопасности [2]: угрозы нарушения конфиденциальности информации; угрозы нарушения целостности информации; угрозы нарушения работоспособности системы. Вредоносные программы нарушают целостность информации, хранящейся в компьютерной системе или передаваемой по каналам связи, направлены на ее изменение или искажение, приводящее к нарушению ее качества или полному уничтожению.

Ситуация усложняется объединением компьютеров в глобальную сеть. Теперь атаки могут проводиться массово и со скоростью урагана на миллионы компьютеров со всего мира, кроме того, свободный доступ программ для проведения атак способствовал появлению так называемых скрипткидисов – людей, использующих готовое программное обеспечение для атаки, даже не всегда понимая принципов его работы.

Вредоносные программы постоянно усложняются и совершенствуются, «Лабораторией Касперского» предложена следующая их классификация [3]:

– Malware – вредоносные программы, созданные специально для несанкционированного пользователем уничтожения, блокирования, модификации или копирования информации, нарушения работы компьютеров или компьютерных сетей. К данной категории относятся вирусы, черви, троянские программы и иной инструментарий, созданный для автоматизации деятельности злоумышленников (инструменты для взлома, конструкторы полиморфного вредоносного кода и т.д.);

– PUPs (Potentially Unwanted Programs) – программы, которые разрабатываются и распространяются легальными компаниями, могут использоваться в повседневной работе, например, системных администраторов. Обладают набором функций, которые могут причинить вред пользователю только при выполнении ряда условий. Например, программы удаленного администрирования, опасны в руках злоумышленника.

Естественно, что с усложнением вредоносных программ совершенствуются и средства борьбы с ними. На рынке присутствует большое количество различных программных средств безопасности, начиная с простеньких антивирусов с сигнатурным поиском и заканчивая комбинированными системами, содержащими в себе, помимо прочего, межсетевой экран с системой предотвращения вторжений, эвристический поведенческий анализатор и так далее.

Чтобы систематизировать существующие защитные технологии, предложено выделять в них два компонента: технический и аналитический [4].

Технический компонент обеспечивает сбор данных, которые, в дальнейшем, анализируются аналитическим компонентом. К этим данным могут относиться: файл, совокупность действий в операционной системе или совокупность эффектов от этих действий. Технический компонент может использовать следующие способы сбора данных:

1. Считывание файлов – самый старый и до сих пор используемый способ. Антивирус побайтово сравнивает поток байт на наличие вирусных сигнатур.
2. Эмуляция кода программы – эмулятор разбирает байтовый код программы на команды и каждую команду запускает в виртуальной копии компьютера.
3. Запуск программы в «песочнице» – является продолжением предыдущего способа, но программе уже позволяется производить контролируемые действия в системе.
4. Мониторинг системных событий – в отличие от предыдущих наблюдает не за конкретной программой, а за всеми производимыми в системе действиями. Технически реализуется перехватом функций ОС.
5. Поиск системных аномалий – анализирует изменения в системе, сравнивая с эталоном или анализируя совокупность отдельных параметров. Для эффективного обнаружения вредоносного кода таким методом необходима сложная аналитическая система – наподобие экспертной системы или нейронной сети, поэтому в настоящее время этот способ разработан мало. Зачатки его можно обнаружить в некоторых антируткит-утилитах, где он реализован на уровне сравнения с определенным срезом системы, взятым за эталон, либо отдельных ее параметров.

Аналитические компоненты по сложности алгоритма принятия решений можно классифицировать следующим образом:

1. Простое сравнение – сравнение единственного объекта с имеющимся образцом. Например, поиск сигнатуры или идентификация подозрительного поведения программы по единственному совершенному ею действию (такому, как, например, запись в критичный раздел реестра).
2. Сложное сравнение – сравнивается один или несколько объектов с шаблонами, а результат является многозначным, каждое значение которого представляется с некоторой достоверностью. Например, определение злонамеренных действий по последовательности вызовов API функций.
3. Экспертная система – решение принимается на основе «тонкого» анализа. Пример: идентификация вредоносного кода по результатам оценки всей накопленной информации о поведении программы, с присвоением каждому из событий веса «потенциальной вредоносности» и расчетом общего результата.

Виды эвристических методов

Далее рассмотрим несколько различных подходов, применяемых в эвристиках.

По определению словаря Ожегова – Шведовой, «эвристика − совокупность исследовательских методов, способствующих обнаружению ранее неизвестного» [5]. Это способ решения проблем в тех случаях, когда не существует формальных методов их решения либо они слишком сложны.

В экспертных системах широко используются эвристики при формализации профессиональных знаний специалистов в той или иной предметной области; в этом случае эвристики описывают, как эксперт обычно решает возникающие в его работе проблемы.

При антивирусной защите некоторые задачи признаны не решаемыми, точнее говоря, невозможно получить точное их решение за ограниченное время [6]. И поэтому единственный выход состоит в использовании эвристик. При этом антивирус детектирует вредоносные программы с помощью анализа их структуры и поведения вместо поиска сигнатур.

Эвристика – это в первую очередь тип аналитического компонента защиты часто с искусственным интеллектом, предназначенного для «нечеткого» способа решения нечетко поставленной задачи. Как эвристические могут рассматриваться аналитические компоненты с экспертной системой или сложным сравнением.

Сильной стороной эвристических методов является возможность поиска новых вредоносных программ (проактивная защита) до того, как будут выделены сигнатуры. Слабые стороны – это высокая вероятность ложных срабатываний, невозможность лечения, низкая эффективность против принципиально новых вредоносных программ [7].

Системы на основе весов и/или правил

В системе, основанной на весах, на начальной стадии каждой найденной подозрительной функциональности присваиваются свои веса, которые затем суммируются. Если сумма оказывается больше некоторого порога, то делается заключение, что программа вредоносная. Но, поскольку, такой подход вызывает частые ложные срабатывания, эта технология по большей части не используется в «чистом» виде.

Сегодня чаще используются подходы, основанные на правилах. При этом сравнивается обнаруженная функциональность с набором правил. Если предопределенное правило либо некоторый их набор соответствуют коду, система сообщает, что возможен вирус [8].

Нейронные сети

Для эвристического обнаружения компьютерных вирусов могут быть применены нейронные сети. Основным материалом для обучения нейронной сети является набор n-грам (последовательностей длиной несколько байтов), который указывает на заражение [9]. Исследователи IBM определили, что наилучший результат достигается при использовании однослойных сетей совместно с системой голосования. В ходе эксперимента обучалась не одна, а множество нейронных сетей, каждая на основе различных характеристик. Чтобы вывести окончательный результат, была использована система голосования, т.е. с позитивным результатом должны были согласиться несколько нейронных сетей. В результате исследования оказалось, что наилучший показатель достигается, когда пять из восьми сетей голосуют за позитивный результат.

На практике нейронные сети очень эффективны против вариантов вирусов, похожих на те, которые использовались в обучающей выборке. Эффективные они и против новых семейств вирусов, которые по поведению достаточно похожи на образцы из обучающей выборки. Исследователи IBM успешно применяют нейронные сети для эвристического определения загрузочных и Win32 вирусов. Например, обученная нейронная сеть успешно обнаруживает сложный полиморфный вирус с неопределенной точкой входа, такой как Zhengxi. Движок нейронной сети, разработанный компанией IBM, представлен в антивирусе от Symantec. Он вырабатывает настолько незначительное количество ложных срабатываний, что используется в основном при сканировании.

Кластерный анализ

Для машинного обучения применимы и другие методы, в частности, кластерный анализ. В [10] описано его применение в случае, когда исходными данными для обучения и сканирования являются сведения о поведении программ в виде последовательности событий, которые сохраняются в базе данных. Каждое событие содержит следующую

информацию: ID события, объект события (например, реестр, файл, процесс, сокет и т.д.), субъект события (процесс, выполняющий действия), вызываемая функция ядра, параметры (например, значение реестра, путь к файлу, IP адрес), статус действия (создан описатель файла, удалён ключ реестра и т.д.). С использованием алгоритма кластерного анализа строится классификация на основе большого количества объектов из базы данных, содержащей последовательности вредоносного поведения; а затем полученные классы применяются при сканировании новых программ. Для определения схожести последовательностей событий применяется алгоритм вычисления расстояния Левенштейна.

Согласованные эвристики

В [11] разные независимые друг от друга эвристики работают согласованно для снижения количества ложных срабатываний:

1. Взвешенная мини-сигнатура – короче обычной сигнатуры и недостаточна для однозначной идентификации вредоносной программы. Однако при использовании совместно с другими мини-сигнатурами риск ложных срабатываний снижается. Каждой мини-сигнатуре назначается свой вес и, когда сумма весов обнаруженных сигнатур достигает 1, программа признается вредоносной.
2. База знаний по вредоносным программам – содержит имена файлов, IP-адреса, адреса электронной почты, CLSID, URL и т.д., используемые во вредоносных программах. Такое решение аналогично базе данных CLSID с сайта http://www.castlecops.com/CLSID.html, но содержит, кроме имен COM-объектов, еще и другие характерные компоненты. Совпадения сканируемых объектов с базой знаний ищутся при помощи регулярных выражений.
3. Нетипичная комбинация функциональности программы. Если программа содержит строки 'SOFTWARE\Microsoft\Windows\CurrentVersion\Run' и 'advapi32.dll', то, вероятно, она добавляет ключ реестра для автозапуска. Если программа содержит 'MIME-Version: 1.0', значит, наверное, она отправляет почту. Соответственно, если присутствует строка 'bank.com', программа может отслеживать URL банка. Каждая из этих функций является обычной, если используются по отдельности, но если они собраны в одной программе, то существует высокая вероятность, что эта программа занимается воровством паролей к банкам и отсылает их по электронной почте.

Существующие решения на базе эвристических методов

На сегодняшний день основными областями применения эвристических методов в борьбе с вредоносными программами являются:

– статический эвристический анализатор – ищет в коде подозрительные признаки (команды, ресурсы и др.), характерные для вредоносных программ;
– динамический эвристический анализатор – эмулирует выполнение программы и наблюдает за наличием подозрительных действий;
– поведенческий блокиратор нового поколения – анализирует последовательность операций, выполняемых в системе.

Чаще всего статический и динамический анализаторы используют совместно.

Большинство антивирусов содержат эвристические методы той или иной сложности для определения неизвестных угроз. Многие антивирусы содержат как минимум статический эвристический анализатор, другие, как NOD32, Антивирус Касперского, BitDefenfer – динамический эвристический анализатор и/или поведенческий блокиратор нового поколения [12].

По результатам различных тестов наиболее эффективными по количеству определенных ранее неизвестных вредоносных программ являются Avira AntiVir Personal Edition Premium, BitDefender, Fortinet, NOD32, AVZ. При этом NOD32 реже других антивирусов показывает ложные срабатывания [10], [13], [14].

Каждый антивирус имеет свои сильные стороны. Так, Антивирус Касперского славится своими декрипторами и оперативно обновляемыми вирусными базами. Эмулятор NOD32 поддерживает практически полный набор инструкций процессора, включая MMX и FPU. Norman Virus Control, помимо виртуального процессора, создаёт «sandbox» (песочницу), эмулирующую реестр и файловую систему [15].

Интеллектуальная система на основе сценариев для обнаружения вредоносных программ. Постановка задачи

Целью статьи является разработка новых подходов для увеличения количества обнаруживаемых вредоносных программ, а также уменьшения ложных срабатываний при их детектировании.

Предлагается экспертная система, основанная на сценариях поведения вредоносного кода, для использования ее при проактивной защите, а именно, в эвристике антивируса и/или поведенческом блокираторе. Для упрощения изложения рассмотрим её в составе эвристика.

Почему должна быть экспертная система?

Почему это должна быть экспертная система, а не обычная программа, например, система для обработки данных? Потому что описание поведения является не данными, а знаниями. В компьютерных науках существуют определения и для данных, и для знаний [16].

Определение 1

Данные – это отдельные факты, которые характеризуют объекты, процессы и явления предметной области, а также их свойства.

Определение 2

Знания – это закономерности предметной области (законы, связи, правила), позволяющие экспертам ставить и решать проблемы в этой области.

Знания преобразуются во время компьютерной обработки следующим образом:

1. Знания в памяти человека как результат мышления.
2. Материальные носители знаний – книги, руководства и т.п.
3. Поле знаний – общепринятое описание базовых объектов предметной области, их атрибутов и правил, а также связей между ними.
4. База знаний на машинных носителях, которая близка к естественному языку – понятному неспециалисту в области компьютерных наук.

Очевидно, что описания поведения вредоносной программы − это знания, а не данные, поскольку в нашем случае мы имеем не только факты, но и правила о том, как работают вредоносные программы.

И поскольку мы имеем дело со знаниями, то для их обработки требуется экспертная система, которая сможет принимать решения о различных компьютерных объектах (проявляют ли они вредоносное поведение или нет), основываясь на базе знаний о поведении вредоносных программ.

Как известно, экспертная система состоит из трех основных частей: базы знаний, системы управления базами знаний, осуществляющей ввод знаний, их хранение, редактирование, добавление и трансляцию во внутреннее представление, а также решателя (машины вывода) для обработки знаний при принятии решений [16].

Существуют различные модели представления знаний, такие как правила (если <условия> то <действия>), семантические сети, фреймы, сценарии (иерархии сценариев), как частный случай фреймов и другие. Таким образом, могут быть и разные системы управления базами знаний для различных типов представления. Также существуют различные способы обработки знаний, в частности, прямой логический вывод, обратный и комбинированный.

Выбор способа представления знаний

Нужно выбрать подходящее представление знаний для описания поведения вредоносных программ. Описание поведения содержит действия, но действия могут быть представлены алгоритмами, правилами или сценариями.

Обычно для описания поведения программ используются алгоритмы. Но в нашем случае необходимо описать поведение не каждой программы, а целых классов вредоносных программ. Кроме того, должна быть возможность описать поведения на разных уровнях иерархии, чтобы пользователю можно было давать объяснения не на уровне элементарных действий, а используя общие доступные понятия. Таким образом, нам нужно некоторое обобщенное представление алгоритмов в иерархическом виде. Для этого предлагается использовать сценарии [16]. И сценарии предпочтительнее правил, поскольку они ближе к алгоритму, нежели правила (могут описывать не только состояния, но и порядок действий, а также итерации).

При этом следует описывать не все поведения, а лишь отдельные характерные фрагменты, по которым можно идентифицировать вредоносные программы.

Представление знаний с помощью сценариев

Рассмотрим, из каких частей состоит сценарий. Сценарии разрабатываются для описания поведения, а каждое поведение обычно преследует какую-то цель. Так, в нашем случае цель − это то, что вредоносная программа пытается сделать, чтобы разрушить систему или ее компоненты (внедриться в систему, заразить бинарный файл и тому подобное). И, таким образом, цель явно или неявно отображается в имени сценария.

Мы должны предсказать, как вредоносная программа могла бы себя повести для достижения цели, чтобы затем при детектировании сравнивать это поведение с ходом работы изучаемой программы. Например, если целью является заражение исполняемых файлов, вирус в первую очередь ищет файлы, открывает их, пишет в секцию кода и модифицирует заголовок. Эти шаги будем называть подцелями.

Подцели − это промежуточные этапы для достижения цели. Таким образом, они могут рассматриваться как части сценария. Каждая подцель может иметь различные условия. Например, «открыть файл» всегда выполняется после «найти файл». Некоторые подцели могут быть обязательными, некоторые – нет. Но такие условия могут быть описаны различными способами.

Некоторые подцели могут иметь дочерние сценарии, описывающие поведение для выполнения подцели. Например, для подцели «записать в секцию кода» мы должны «перейти в секцию кода», «использовать API-функцию WriteFile» и так далее. Таким образом, в результате построений получается иерархия сценариев.

Далее опишем и приведем несколько примеров, описывающих поведение вредоносных программ, и, таким образом, получим набор иерархий сценариев. Первая иерархия сценариев показана на рис. 1.

Цель сценария верхнего уровня − заразить исполняемый файл, то есть внедрить вирус таким образом, чтобы инфицированная программа, с одной стороны, оставалась работоспособной, а с другой – служила источником для размножения вируса.

Подцели могут быть двух типов: 1) базовые (не имеющие дочерних сценариев) и 2) подцели, определенные на основе базовых. Базовые подцели соответствуют тем действиям, которые мы перехватываем в системе, например, вызов API функции. Таким образом, двигаясь от цели, мы определяем ее подцели, затем эти подцели через другие и так далее до тех пор, пока все подцели прямо или косвенно будут определены через базовые. Для первой иерархии сценариев, представленной на рис. 1, базовыми будут следующие подцели: API функция FindFirstFile, API функция FindNextFile, API функция WriteFile, API функция CreateFile и API функция OpenFile. Некоторые типы базовых подцелей и их формальные описания перечислены ниже.

Таким образом, каждый сценарий из иерархии состоит из имени, отображающего цель выполнения сценария, и списка подцелей.

Теперь необходимо определить, как формально записывать сценарии, и какие типы подцелей будут базовыми. Для записи сценариев мы предлагаем использовать язык, аналогичный языку регулярных выражений. Рассмотрим три базовых операции и две производные от них. Базовые операции:

1. Конкатенация. Обозначается дефисом «–». *a* − *b* означает последовательное выполнение подцелей, сначала *a*, затем *b* (например, Find File, а за ним Open File).

2. Дизъюнкция (ИЛИ). Обозначается знаком «|». *a* | *b* означает, что выполняется *a* или *b*.

3. Итерация. Обозначается знаком «*». *a** означает, что сценарий а может повторяться произвольное число раз (в том числе и нулевое).

Пример

*(a | b)** − произвольное число раз повторяется *a* или *b*, т.е. любая комбинация из символов *a* и *b*, например, *aaab*, *bbbb*, *ba*, и т.п.



Рисунок 1 – Иерархия сценариев для описания поведения паразитического вируса

Производные операции:

1. ? – не обязательная подцель, имя завершается символом «*?*».

2. + подобно *, но повторения 0 раз не допускаются, т.е. разрешено только одно или более повторений.

Пример

b−*a*? −*c* − здесь возможны два варианта: *bac* или *bc*.

Теперь мы можем записать формально несколько поведений, характерных для вредоносных программ, и, таким образом, получим набор иерархий сценариев. Пример для первой иерархии сценариев:

Сценарий работы паразитического вируса = (Поиск_исполняемого_файла – Открытие_файла – Запись_в_секцию_кода – Модификация_заголовка?)+

Поиск_исполняемого_файла= API_функция_FindFirstFile–API_функция_FindNextFile?

Запись_в_секцию_кода = Переход_к_секции_кода – API_функция_WriteFile

Открытие_файла = API_функция_CreateFile | API_функция_OpenFile

API-функции являются базовыми подцелями и в дальнейшем не уточняются. Они соответствуют событиям, перехваченным анализирующей системой (базовым подцелям).

Может возникнуть вопрос, почему недостаточно хранить лишь сценарии низших уровней иерархии, где находятся только базовые подцели? Ответ можно сформулировать следующим образом: если хранятся только сценарии с базовыми подцелями, то их «понимание остается только в голове у создателя».

В случае же иерархии на верхних уровнях располагаются понятия, являющиеся обобщениями понятий нижележащих уровней. И, таким образом, мы получаем систему, которая может «понимать», что происходит, и объяснить это в терминах сценариев вышележащих уровней с нужной степенью детализации.

Расширение возможностей (совершенствование) языка сценариев

Для более точного анализа и чтобы избежать ложных срабатываний нам понадобятся переменные и параметры подцелей. Например, необходимо описать поведение вредоносной программы, «прописывающей» себя в Автозагрузке (рис. 2). Цель этого сценария − внедрение вредоносного кода в систему.

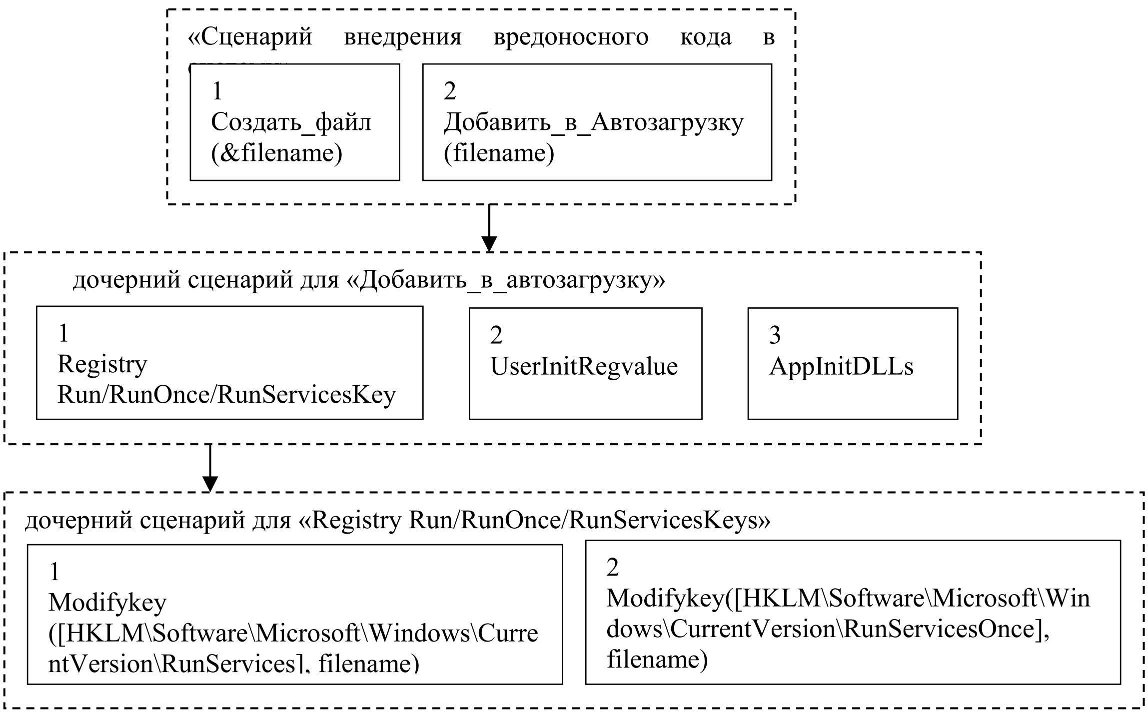


Рисунок 2 – Иерархия сценариев для внедрения вредоносного кода в систему

Итак, файл записывается на диск, а затем его имя «прописывается» в реестре. Мы хотим проверить не только, что произошли два события – запись файла на диск и изменение в реестре, – но также убедиться, что имя файла для обоих событий одно и то же. Это можно на нашем языке формально описать следующим образом:

Create_File (&filename) –ModifyKey(HKLM\Software\Microsoft\Windows\CurrentVersion\Run, filename),

где *filename* – переменная, которая передается первой подцели по ссылке, а второй – по значению. Такая запись означает, что первая подцель считается выполненной, когда создается любой файл, и его имя запоминается в переменной *filename*. А вторая подцель выполняется тогда и только тогда, когда изменяется ключ реестра HKLM\Software\Microsoft\Windows\CurrentVersion\Run, и туда записывается информация из переменной *filename*.

Таким образом, подцели могут иметь параметры двух типов*:* по ссылке (мы можем описать с помощью «&») и по значению (по умолчанию).

Мы описали две иерархии сценариев и теперь можно перечислить типы подцелей (встретившиеся в примерах), которые анализирующая подсистема должна перехватывать (т.е. базовые подцели).

Список базовых подцелей:

1) модификация ключа реестра *ModifyKey(keyname, data)*,

2) вызов API-функции *API_<function name>(parameters)*,

3) запуск процесса *Run(&processname)* или *Run(processname)*.

Теперь мы можем дать расширенное определение «Сценария внедрения вредоносного кода в систему»:

Внедрение_вредоносного_кода_в _систему = Создать_файл (&filename) – Добавить_в_автозагрузку (filename)

Создать_файл (&filename) = API_функция_CreateFile(..,&filename,..) | API_функция_OpenFile(..,&filename,..)

Добавить_в_Автозагрузку (filename) = Regestry_Run_RunOnce_RunServices_Keys (filename)

Regestry_Run_RunOnce_RunServices_Keys = Modifykey([HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices], filename)
| Modifykey([HKLM\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce], filename)

Архитектура эвристика

Детектирование вредоносного кода с использование эвристика проходит три фазы [17]: фаза декодирования, фаза исследования и фаза оценки. Первые две фазы относятся к техническому компоненту, последняя – к аналитическому.

Предназначение фазы декодирования – эмуляция требуемого количества инструкций, необходимых вирусу для декодирования своего тела.

Предназначение фазы исследования – эмуляция, по крайней мере, единожды всех доступных в программе секций кода, которые предположительно могут содержать вирусы.

Предназначение фазы оценки – анализ любых подозрительных действий, которые были найдены во время декодирования и исследования, для определения инфицированности программы. Это фаза делится на две.

Первая – составление списка всех наблюдаемых поведений программы с использованием статического и динамического подходов.

Вторая – анализ выявленных поведений. Именно здесь определяется, является ли множество обнаруженных поведений свойственным вредоносным программам или нет. Для этого используется база знаний.

Фаза декодирования, исследования и первая часть фазы оценки выходят за рамки исследований, описанных в этой статье. Именно вторая часть фазы оценки будет эксперт-

ной системой, основанной на сценариях, то есть это будет решатель ЭС, который на основе логических выводов сможет проанализировать и сравнить просканированные файлы (их вредоносное поведение) со сценариями для каждого типа вредоносных программ или со сценариями различных типов вредоносного поведения.

Полная архитектура эвристика отображена на рис 3. Она состоит из двух подсистем:

1. Система управления базами знаний с базой знаний, базирующейся на иерархиях сценариев вредоносного поведения. Система управления базами знаний предназначена для организации ввода знаний, их хранения, редактирования, добавления и перевода во внутреннее представление.

2. Эвристический сканер, включающий решатель экспертной системы, необходим для определения инфицированности исследуемого объекта.

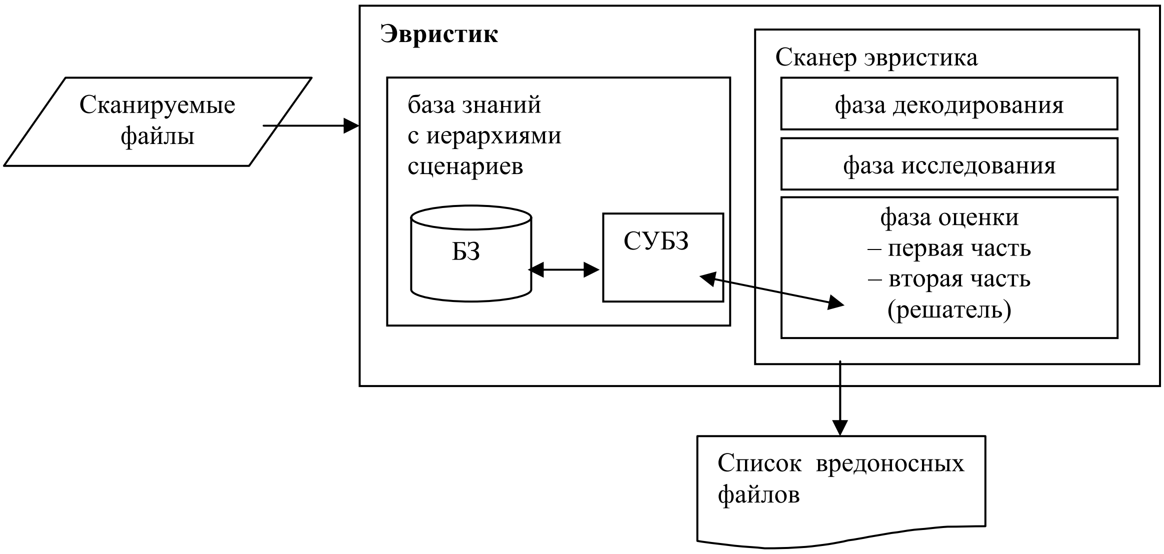


Рисунок 3 – Архитектура эвристика, использующего экспертную систему

Заключение

Рассмотрим преимущества предлагаемой технологии.

Экспертная система в большей степени, чем традиционные программы, подходит для решения задачи анализа при детектировании вредоносных программ по следующим причинам:

1. Знания хранятся не в программном коде, где они могут быть изменены только программистом, а в базе знаний, и таким образом, могут быть просмотрены, изменены и добавлены экспертом – вирусным аналитиком.

В нашем случае правила для эвристика записаны в текстовом файле с использованием специального языка. Множество правил может быть исправлено и расширено без перекомпиляции эвристика, таким образом, эвристик сам по себе универсален, т.е. независим от множества правил.

2. Экспертная система может объяснить свои решения, опираясь на логические рассуждения и иерархию сценариев. Для этих целей мы можем задать соответствующие сообщения для подцелей, которые будут показаны пользователю.

Сценарии вредоносного поведения имеют преимущества по сравнению с другими моделями представления знаний, так как они могут описать всё то же, что и другие технологии и, к тому же, дополнительную информацию:

1. На основе сценариев можно легко строить правила, которые учитывают не только вредоносные действия сами по себе, но также их множества и порядок исполнения.

2. Мы можем использовать переменные, которые будут связывать действия и, таким образом, гарантировать с высокой вероятностью, что эти действия не случайны, и что они направлены на достижение вредоносных целей.

3. Сценарии более емкие, чем другие представления. Например, пусть задан сценарий, который имеет две подцели, причем, первая может быть выполнена тремя способами, вторая – четырьмя. В этом случае мы имеем двенадцать вариантов поведения.

Такой подход при построении антивируса более эффективен, чем традиционные, без использования интеллектуальных технологий, так как позволяет увеличить количество детектируемых вредоносных программ и уменьшить количество ложных срабатываний за счет более «тонкого» анализа приближенного к работе человека-эксперта.

Литература

1. Александр Евангели. Куда направлен антивирусный вектор // BYTE. – 2005. – № 12. – С. 74-76.
2. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях / Под ред. В.Ф. Шаньгина. – М.: Радио и связь, 1999. – 328 с.
3. Юлия Кряквина. Шестой саммит вирусных аналитиков «Лаборатории Касперского»: репортаж с открытого пресс-дня. – Режим доступа: http://www.ixbt.com/cm/kaspersky-summit2k7.shtml
4. Алиса Шевченко. Технологии обнаружения вредоносного кода. Эволюция. – Режим доступа: http://www.viruslist.com/ru/analysis?pubid=204007574
5. Ожегов С.И., Шведова Н.Ю. Толковый словарь русского языка: 72500 слов и 7500 фразеологических выражений / Рос. АН, Ин-т рус. яз., Рос. фонд культуры. – М.: Азъ, 1992. – 955 с.
6. Understanding Heuristics: Symantec's Bloodhound Technology. – Режим доступа: http://securityresponse.symantec.com/avcenter/reference/heuristc.pdf.
7. «Лаборатория Касперского». Антивирусная защита компьютерных систем. – Режим доступа: http://www.intuit.ru/department/security/antiviruskasp/6/
8. Markus Schmall. Heuristic Techniques in AV Solutions: An Overview. – Режим доступа: http://www.securityfocus.com/infocus/1542
9. Peter Szor. The Art of Computer Virus Research and Defense, Addison Wesley Professional, – p. 744.
10. Tony Lee & Jigar J. Mody. Microsoft Corp. Behavioral Classification. / Presented on the EICAR Conference – May, 2006. – Режим доступа: http://www.microsoft.com/downloads/details.aspx?FamilyID=7b5d8cc8-b336-4091-abb5-2cc500a6c41a&DisplayLang=en
11. John D. Park. Symantec Security Response, USA. Cooperative heuristics. – Режим доступа: http://www.virusbtn.com/virusbulletin/archive/2006/01/vb200601-coop-heuristics#id3152982
12. Different classes of security software. From CastleCopsWiki. – Режим доступа: http://wiki.castlecops.com/Different_classes_of_security_software
13. Анатолий Ализар. Эвристика эффективнее, чем обновление антивирусных баз. – Режим доступа: http://www.compdoc.ru/secur/virus/heuristics/
14. Сергей Ильин. Сравнение эффективности проактивной антивирусной защиты. – Режим доступа: http://www.anti-malware.ru/index.phtml?part=compare&anid=proactive1
15. Крис Касперски. Обманчивый антивирус. – Режим доступа: http://1050049.ru/home.asp?artId=5389
16. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – СПб.: Питер, 2001. – 384 с.
17. Nashenberg, Caryu S. The Method of Dynamic Heuristic for Computer Viruses Detection Using Decoding and Research Phases. – Режим доступа: http://www.nf-team.org/drmad/zf/zf5/zf5_011.htm

В.М. Рувинська, Е.Л. Беркович, А.А. Лотоцький

Евристичні методи дуфектування шкідливих програм на основі сценаріїв

У статті розглянуті існуючі методи боротьби з шкідливими програмами, а також запропонований інтелектуальний метод їх детектування на основі сценаріїв. Сценарії дозволяють представити поведінку шкідливих програм в ієрархічному вигляді; наведений приклад сценарію поведінки паразитичного вірусу і сценарію впровадження кода в систему. Запропоновано формально описувати сценарії на основі регулярних виразів. Розроблена архітектура евристика, що містить експертну систему на основі сценаріїв.

V.M. Ruvinskaya, Ye.L. Berckovich, A.A. Lotoskiy

Heuristic Method of Malware Detection on the Basis of Scripts

In the article we consider existent methods of fight against a malware, and offer the scenarios based intellectual method of their detection. Scenarios allow presenting any malware behaviour in a hierarchical kind; virus and system intrusion scenarios examples are presented. It is suggested to describe scenarios on the basis of regular expressions. Heuristic architecture was developed, that containing scenarios based consulting model.

Статья поступила в редакцию 18.07.2008.